Power consumption behaviour of multiplier block algorithms.

Suleyman Demirsoy
Andrew Dempster
Izzet Kale

Cavendish School of Computer Science

POWER CONSUMPTION BEHAVIOUR OF MULTIPLIER BLOCK ALGORITHMS

Süleyman Sırrı Demirsoy, Andrew G. Dempster and Izzet Kale

Applied DSP and VLSI Research Group, Department of Electronic Systems
University of Westminster, 115 New Cavendish St, London, W1W 6UW, UK
Tel: +44 20 7911 5000 - 3611(ext) e-mail: demirss@cmsa.wmin.ac.uk

ABSTRACT

Multiplier blocks have been used primarily for the reduction of circuit complexity. Using new algorithms, it has been shown that they can also be used for effective reduction of power consumption in digital filter circuits. In this paper, the new GP score method is used as a relative power measure to compare digital filter multiplier blocks designed using the BHM, RAGn and C1 algorithms.

1. INTRODUCTION

Digital filter implementations employing multiplier blocks (MB) offer substantial reduction in the multiplier area [2] and power consumption [3]. This technique is based on sharing the intermediate terms that are generated by addition, subtraction and shifting of the input sample between different coefficients. By the clever combination of these partial products it is possible to form the products with comparably few adders/subtractors [1].

There are algorithms to facilitate the design of multiplier blocks. These algorithms take the coefficient set of the filter as the input and produce a set of partial products that leads to the generation of the final products in an optimal way defined by the criteria of the algorithms. The BHM algorithm [2] looks for the closest match to the target value from a range of multiples of all fundamentals and adds to graph and continue to do so until every coefficient value is formed. The RAG-n algorithm tries to find a set of partial products with the fewest adders by firstly including all partial products that require a single adder, then examining whether any further products can be produced when adding only a single adder to the graph. This algorithm operates for coefficient word-lengths up to 12-bits [2] The C1 algorithm, which employs RAG-n as a sub-algorithm, aims to find a multiplier block with the shallowest possible adder chain from the input to any output [3]. It starts by including in the designed graph all numbers that require a single adder, i.e. "cost 1". This produces a very shallow graph. Cost 1 numbers are then removed from the graph if this does not result in an increase in the graph's depth. The ultimate goal in any of these algorithms is the reduction in area and/or power usage.

In digital CMOS circuits the major source of power dissipation is due to transitions at the circuit nodes, and it is formulated as follows [5]:

$$P_{switching} = \alpha_{0\to 1} C_L V_{dd}^2 f_{clk} \qquad (1)$$

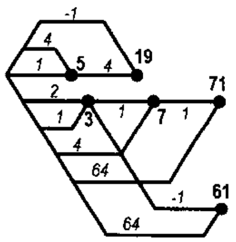
where $\alpha_{0\to 1}$ is the node transition activity factor (the average number of times the node makes a transition in one clock period), C_L is the load capacitance, V_{dd} is the supply voltage to the circuit and f_{clk} is the clock frequency of the circuit. It can be easily inferred that the number of transitions taking place in two circuits is an acceptable measure for the comparison of the relative power consumption provided that the number of nodes are not significantly different [5].

Recently, a high level method to estimate the relative transition activity of multiplier blocks has been proposed [4]. "GP Score" shows good correlation with the actual transition counts at each node in a multiplier block for Virtex FPGA implementations. The algorithm is applicable to any multiplier block in general, however, the MATLAB script [6] operates on a table of numbers that represents the details of the multiplier block in Dempster format as detailed in the following section.

In this paper, we use the GP Score as the metric to compare the relative power of the BHM, RAG-n and C1 algorithms. Section 2 describes the details of the experiments performed. Section 3 discusses the simulation results. Section 4 comments on the behavior of the algorithms and Section 5 concludes the paper.

2. ANALYSIS

The multiplier block algorithms mentioned above, are coded in MATLAB and generate multiplier blocks in a table format composed of the partial results and the input connections and shift values for each adder/subtractor in the multiplier block. Figure 1a shows a signal flow graph for a multiplier block composed of six partial results/coefficients. Each vertex (●) represents an adder with two inputs. The numbers on the edges



(a)

Partial results	Input1	Input2	Edge1	Edge2	Logic depth
5	1	1	1	4	1
3	1	1	1	2	1
7	1	3	1	2	2
19	1	5	-1	4	2
71	1	7	64	1	3
61	1	3	64	-1	2

(b)

Figure 1 A multiplier block in (a) signal flow graph form, (b) in Dempster format

0-7803-7523-8/02/$17.00 ©2002 IEEE

are the values that the inputs are multiplied by. These multiplications are realized by shifting the input by hard wiring. Negation is realized by replacing the adder with a subtractor.

Figure 1b is the corresponding table in Dempster format for the signal flow graph given in Figure 1a. Each partial product is given in a single row and characterized by the input connections and scaling factor. A '1' that appears in the input (2nd and 3rd) columns means the graph input signal itself is connected to the adder. A partial product given in the n^{th} row can use the partial products that appear in the upper rows (1 to n-1) as the inputs. The edge values are all powers of 2 and correspond to a shift of $\log_2$ bits of the given number. The last column of the table is the indicator of the maximum number of adders on the path from the input to that partial result.

It is of interest to know how the algorithms perform in terms of power on average for certain word-length and number of coefficients. Their application to different coefficient sets would result in different behaviors. Hence, an investigation over a large number of random coefficient sets for different word-lengths and different set sizes is required.

A set of experiments identifies the behavior of the algorithm for a fixed coefficient set size of 25 and coefficient word-length of 8-bits, 10-bits and 12-bits for an 8-bit input signal over 100 random coefficient sets. Another experiment observes the algorithms over a fixed coefficient word-length of 12-bits for various set-sizes of 10, 15, 20 and 25 for 100 random sets.

To get an idea of the practical use of the algorithms on a variety of real filter implementations, another set of experiments was performed over a set of Low-Pass (LP), Band-Pass (BP) and High-Pass (HP) filters.

All experiments were performed in the MATLAB environment. The random filter sets are generated using the `random` command. Real filter implementations were done by the `remez` command that generates linear phase FIR filters for the given frequency behavior.

3. RESULTS

Individual GP Scores of the multiplier-blocks generated by three different algorithms for filters with random coefficient sets are shown in Figure 2. Figure 2a is for 25 coefficients of 10-bit word-length and Figure 2b is for 15 coefficients of 12-bit word-length. The goal of the analysis was to find out the GP Score characteristics of each algorithm for different word-length and set size specification. It is observed that, for any particular coefficient set, C1 generates the lowest GP Score among the algorithms. The behavior of RAG-n and BHM varies greatly for different sets. Despite their GP Scores being similar on average in Figure 2a, the standard deviation of the individual scores is greater for RAG-n. On the contrary, in Figure 2b, BHM produces more diverse results and on average has the higher GP Score.

Figures 3a and 3b display the average GP Scores and standard deviations for random coefficient sets of various sizes and word-lengths. In Figure 3a, the results are classified in terms of the set size for the three algorithms where the word-length of the coefficients are kept constant at 12-bit. As expected, the GP Score increases with the increased coefficient set size. This is because the multiplier block gets larger, and can hence be expected in general to consume more power. The C1 algorithm generates the multiplier blocks with the lowest GP Score over all set sizes and has the lowest standard deviation for the random examples. BHM gives the highest average GP Score for set sizes of up to 25 coefficients and has the highest standard deviation among its results. RAG-n produces a higher GP Score for 25 coefficients although it still has lower standard deviation.

Figure 3b shows the average GP Scores of random coefficient sets for a fixed set size of 25 and varying word lengths. The C1 algorithm consistently produces the lowest GP scores with the minimum deviation. For shorter word-lengths, RAG-n and C1 mostly generate the same multiplier blocks for a given coefficient set. However, the variation in the outcomes of RAG-n is the highest among all. BHM gives better results than RAG-n as the word-length increases, however, the deviations in the results are substantially higher.

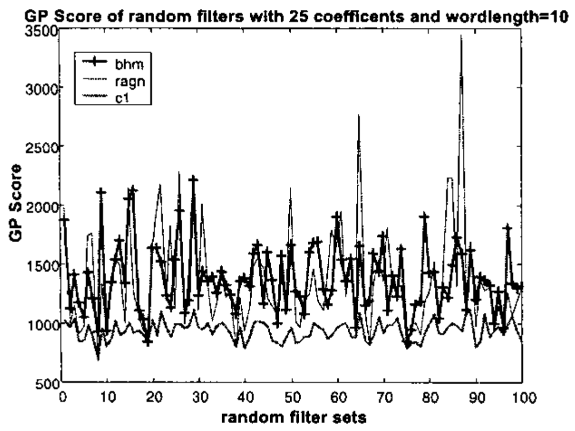
Table 1 displays the GP Scores and adder counts of the multiplier blocks generated by the three algorithms for different types of real filters. There are two lowpass filters, two highpass filters and two bandpass filters. Each filter has a different pass-band and stop-band specification. The coefficients generated by the Matlab `remez` function are normalized to have the maximum value of 1 and then multiplied by 4096 and rounded using MATLAB's `floor` command to convert them to 12-bit integers. The three algorithms are then run to generate the multiplier blocks.

As expected from the previous experiments, C1 gave the lowest GP Score despite the increased adder count in examples 1, 2, 5and 6. BHM resulted in the highest GP Scores for all cases. For 12-bit word-length and 25 coefficients RAG-n performed worse than BHM, however in the examples given here, most of the coefficients are less than 12-bit numbers as can be deduced from Figure 4. In this figure, absolute values of the magnitudes of the coefficients for some filters from Table 1 are shown. The magnitudes of the coefficients of the HP filters are comparably lower than LP and BP filters resulting in the lowest GP Scores of the three types of filters. BP filters have more coefficients with bigger magnitude and have higher GP Scores.

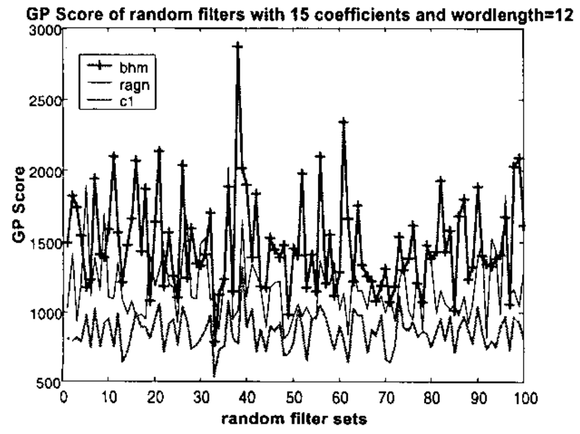
Although the results we have reported for the various filters are indicative of the HP filter resulting in the smallest multiplier blocks due to the density of the small coefficient magnitudes, this is in no way a general characteristic for the HP filters only. On the contrary we could have similar coefficient densities for any filter response. The main factor that primarily decides the coefficient magnitude densities in practical filter implementation (predominantly FIR) is the active pass-band bandwidth.

4. DISCUSSION

Figures 2 and 3 clearly show that not only does the C1 algorithm produce designs that are likely to consume less power than the other algorithms, but also that we can have more confidence in those designs, because the variation in the GP Score is lower. To some extent, this high degree of variation in BHM and RAG-n is due to the fact that they were designed as algorithms to reduce adder count, and the power implications were ignored.

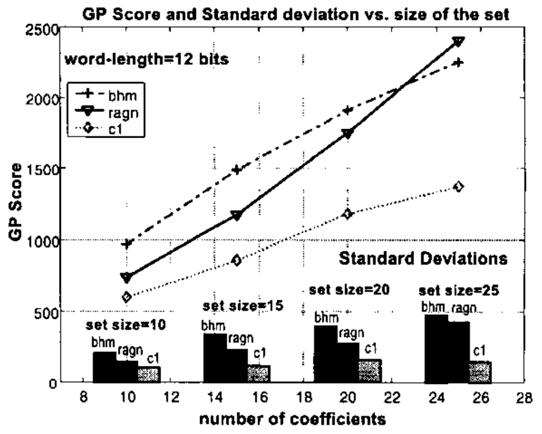


(a)

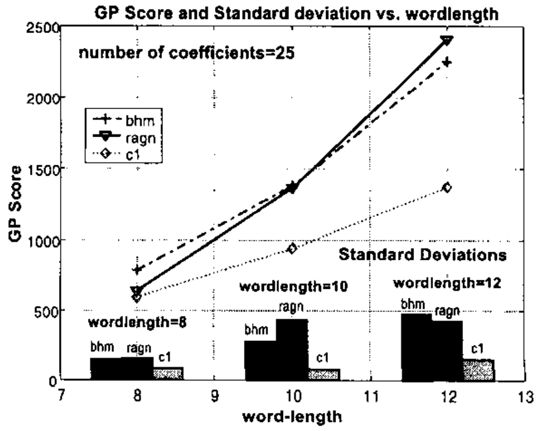


(b)

Figure 2 GP Scores of the multiplier blocks produced by different algorithms for 100 random coefficient filters (a) word-length=10, set size=25 (b) word-length=12, set size=15



(a)



(b)

Figure 3 Average GP Scores and standard deviations of the multiplier blocks produced by different algorithms over 100 random coefficient filters (a) different set-sizes, word-length=12 (b) different word-lengths, set size=25

It has been noted that when the RAG-n algorithm is operating on a coefficient set that has no "cost-1" coefficients, heuristics must be used when producing the multiplier block [2]. This causes problems: the algorithm runs for much longer, and it produces graphs that are not guaranteed to be optimal. The C1 algorithm avoids these problems by including as many cost-1 coefficients as possible. However, it leaves the C1 algorithm vulnerable to the obvious next layer of problems at cost 2.

For example, consider the coefficient set [3 2123 3746 1689], which has individual multiplier costs [1 3 3 4]. RAG-n would design a 9-adder graph with depth 5 and GP Score 316. There is only one adder at depth 5 and it (1873 = 3746/2) is the last inserted into the graph, as a combination of 1 x 1689 (at depth 4) and 8 x 23 (depth 2). The 23 is inserted because it is the minimum number that would allow us to get to 1873 using one adder. However, if instead, 2049 was used and we had 1873 = 2049 – 8 x 11, then this adder would have depth 3, the overall graph would have depth 4 and the GP Score would reduce to 267 from 316. In other words, a relatively arbitrary decision has made a significant difference to the power consumption of the design.

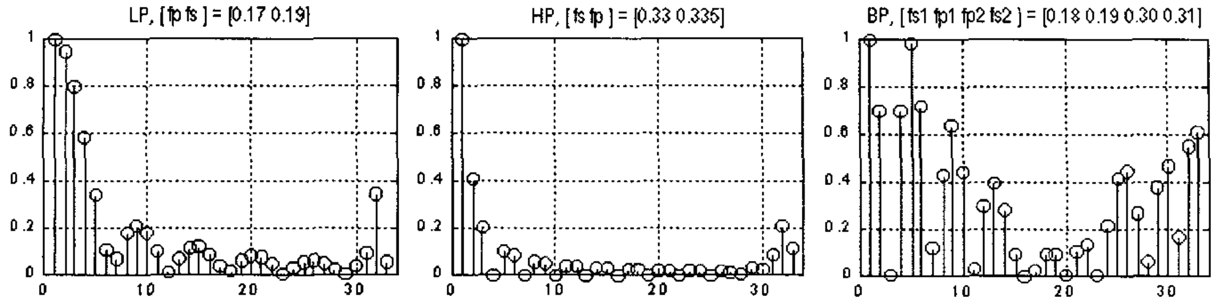
Another problem we noted while analyzing the results was that C1 did not remove cost-1 coefficients that could *reduce* logic depth. Obviously, there is more to designing shallow graphs than our original simple-minded assumption (the premise of the C1 algorithm) that an abundance of cost-1s will guarantee minimum depth.

5. CONCLUSION

The main conclusion of the paper supports our original hypothesis: that the C1 algorithm delivers lower-power designs than the BHM or RAG-n algorithms. It also produces a more

TABLE 1 GP Scores and Adder Counts for a set of LP, HP and BP linear phase FIR filters

No	FILTER SPECIFICATIONS	GP SCORES			ADDER COUNT		
	Coefficient word-length=12-bit	BHM	RAG-n	C1	BHM	RAG-n	C1
1	Order=64, LP, [fp fs]=[0.20 0.21]	1305	1195	1068	31	30	32
2	Order=64, LP, [fp fs]=[0.17 0.19]	1666	1282	1047	29	29	32
3	Order=64, HP, [fs fp]=[0.34 0.36]	941	716	716	24	24	24
4	Order=64, HP, [fs fp]=[0.33 0.335]	860	699	699	23	22	22
5	Order=64, BP, [f_{s1} f_{p1} f_{p2} f_{s2}]=[0.18 0.19 0.30 0.31]	1840	1619	1165	33	32	34
6	Order=64, BP, [f_{s1} f_{p1} f_{p2} f_{s2}]=[0.20 0.21 0.35 0.37]	2011	1793	1342	33	31	34



(a) (b) (c)

Figure 4 Normalized coefficients magnitudes of the (a) 2nd set (b) 4th set (c) 5th set from Table 1. The distribution of the 64 coefficients is symmetric in all filters in Table 1, therefore only half of the set is given in the figures.

reliably consistent power consumption for a given random specification. RAG-n seems to work better than BHM for short word lengths and smaller coefficient sets.

However, the C1 algorithm is built on the RAG-n algorithm, which has been seen to behave erratically. In fact, this study seems more useful in pointing the way to a new low-power algorithm, which could modify C1 as follows:

i) Use both RAG-n and BHM at each step and keep the better result
ii) Allow the logic depth of the algorithm to decrease as cost-1 coefficients are eliminated
iii) Allow the number of adders to increase when logic depth is decreased
iv) Consider using only the "optimal" part of RAG-n and if there are coefficients left, use other heuristics, such as the introduction of cost 2 coefficients, or the addition into the graph of some of the required coefficients ±1, ±2, etc.
v) Investigate re-designing RAG-n to ensure that when it has a choice of new adders to include, it chooses the one at minimum depth.
vi) Use GP Score as the criterion for "best graph" rather than logic depth

Another alternative would be to approach the C1 problem the other way around, i.e. to add a single cost-1 and see if it helps with cost or depth or GP Score. Then continue adding cost-1s till no further progress is made.

6. ACKNOWLEDGEMENT

The authors would like to thank Mark Zufferey for his generosity in allowing us to use his resources for the work of this paper.

7. REFERENCES

[1] Bull D. R. and D H Horrocks, "Primitive operator digital filters", IEE Proceedings G, vol 138, no 3, pp401-412, Jun 1991

[2] A G Dempster and M D Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters" IEEE Trans Circuits and Systems II, vol 42, no 9, pp569-577, September 1995

[3] Dempster A. G., S. S. Demirsoy, I. Kale, "Designing Multiplier Blocks with Low Logic Depth", *IEEE Int. Symp. on Circuits and Systems* (ISCAS'2002), vol. 5, pp. 773-776, Arizona, US, May 2002

[4] Demirsoy S. S., A. G. Dempster and I. Kale, "Power Analysis of Multiplier Blocks", *IEEE Int. Symp. on Circuits and Systems* (ISCAS'2002), vol.1, pp.297-300, Arizona, US, May 2002

[5] Chandrakasan A.P. and R.W. Brodersen, "Minimising power consumption in digital CMOS circuits", *Proc. IEEE*, vol. 83, pp. 498-523, April 1995.

[6] http://www.cmsa.wmin.ac.uk/~demirss/gpscore